DiyI0t

MQ2 Gas Sensor Tutorial for Arduino, ESP8266 and ESP32

In this tutorial I show you how to use the MQ2 gas sensor with your Arduino, ESP8266 or ESP32 microcontroller.

After we dive into the functionality of gas sensors in general, we build a gas alarm with the MQ2 gas sensor that detects smoke and turn on an active buzzer as alarm.

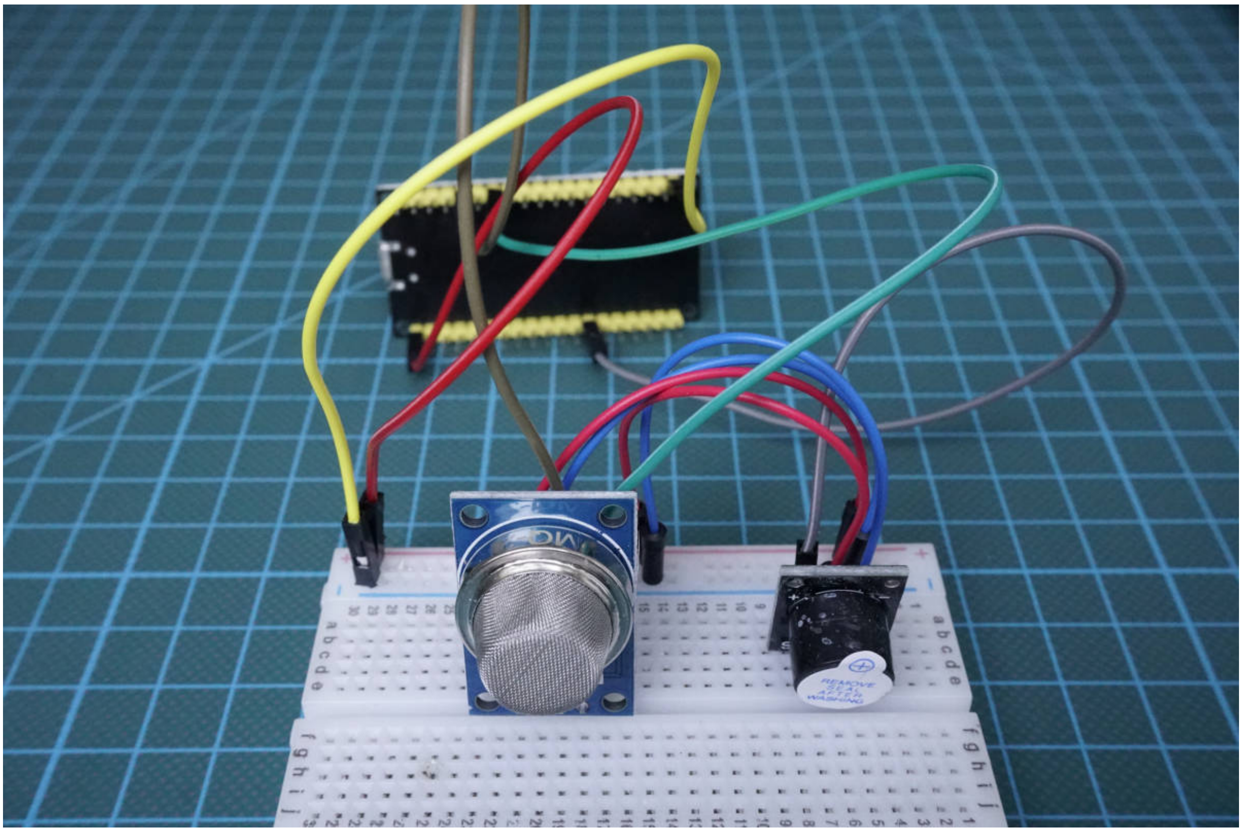
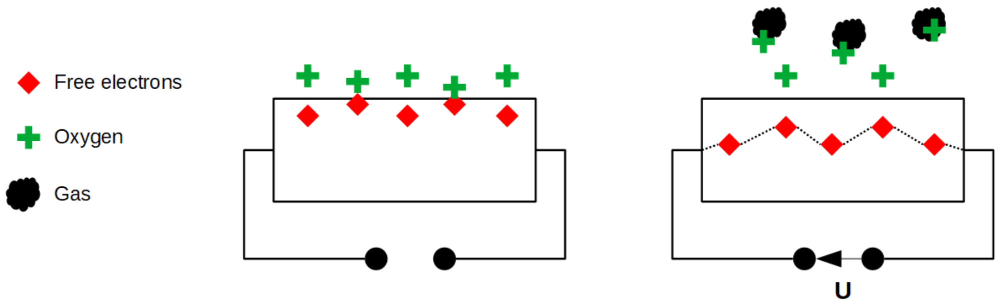


Table of Contents

General functionality of Gas Sensors

First of all we start with the general functionality of gas sensors. Inside the gas sensor is a chemiresistor that changes the resistance based on its sensing material. The following picture helps us to understand this functionality.



In most cases the sensing material is a tin dioxide (SnO2) material that has free electrons inside. These free electrons are attracted by the oxygen towards the surface of the sensing material (left side of the picture). On the surface, the oxygen is absorbed, due to the heated surface and therefore there are no free electrons in the tin dioxide. The result is: without free electrons there is no electrical current flow.

In an environment of toxic or combustible gases, the gas breaks the connection between the absorbed oxygen and the electrons. (right side of the picture) The released electrons are now free and back in their initial position, where they enable the current flow. How high the current flow is, depends on the amount of free electrons available in the SnO2 that is proportional with the concentration of toxic or combustible gases.

Due to Ohms law a higher current flow results in a higher potential difference, which is measured as output voltage on the analog output of the sensor, using a simple voltage divider network. Therefore the concentrations of gas can be measured.

The type of sensor is defined by the sensing material inside the sensor. Although some of the gas sensors are sensitive to multiple gases, **the sensor can not identify which of the gases are in higher concentration**.

Overview of Gas Sensors for Microcontroller

The following table gives you an overview about the different gas sensors that are available and what gasses they are able to detect. I concentrated on the most used gas sensors, because there are more available then shown in the table.

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**135**	**214**	**216**	**303A**	**306A**	**307A**
Methane	x		x							x				
Butane	x				x								x	
LPG	x			x	x								x	
Smoke	x	x										x		
Alcohol		x										x		
Ethanol		x										x		
CNG Gas			x											
Natural Gas				x						x	x			

	2	3	4	5	6	7	8	9	135	214	216	303A	306A	307A
Carbon Monoxide						x		x						x
Hydrogen Gas							x							
Coal Gas								x			x			
Liquefied Gas								x						
Air Quality									x					
Flammable Gas														

MQ2 Gas Sensor Module Overview

In this tutorial I use the MQ2 gas sensor module to detect the butane that is emitted by a lighter. Before we start using the MQ2 gas sensor, we have to know how the sensor itself and the whole module with all other electronic components is working. Therefore we start with the technical datasheet of the whole module and how the MQ2 gas sensor is measuring the gas concentration before we take a closer look at the electronic components on the senor module as well as the pinout of the MQ2 gas sensor module.

MQ2 Gas Sensor Module Technical Datasheet

The following table shows the technical datasheet for the MQ2 gas sensor module.

Operation Voltage	4.5V...5V
High Gas Sensitivity	Methane, Butane, LPG, Smoke
Concentration	300ppm...10,000ppm
Power Consumption	88mA @5V = 440mW
Preheat Duration	2 Minutes

The MQ2 gas sensor has an **operating voltage between 4.5V and 5V** that is also used as heater for the surface of the SnO2. Therefore the microcontroller we use has to provide a stable 5V output. Arduino microcontroller have already an operation voltage of 5V but we have to power the ESP8266 and ESP32 boards via USB to get an 5V output.

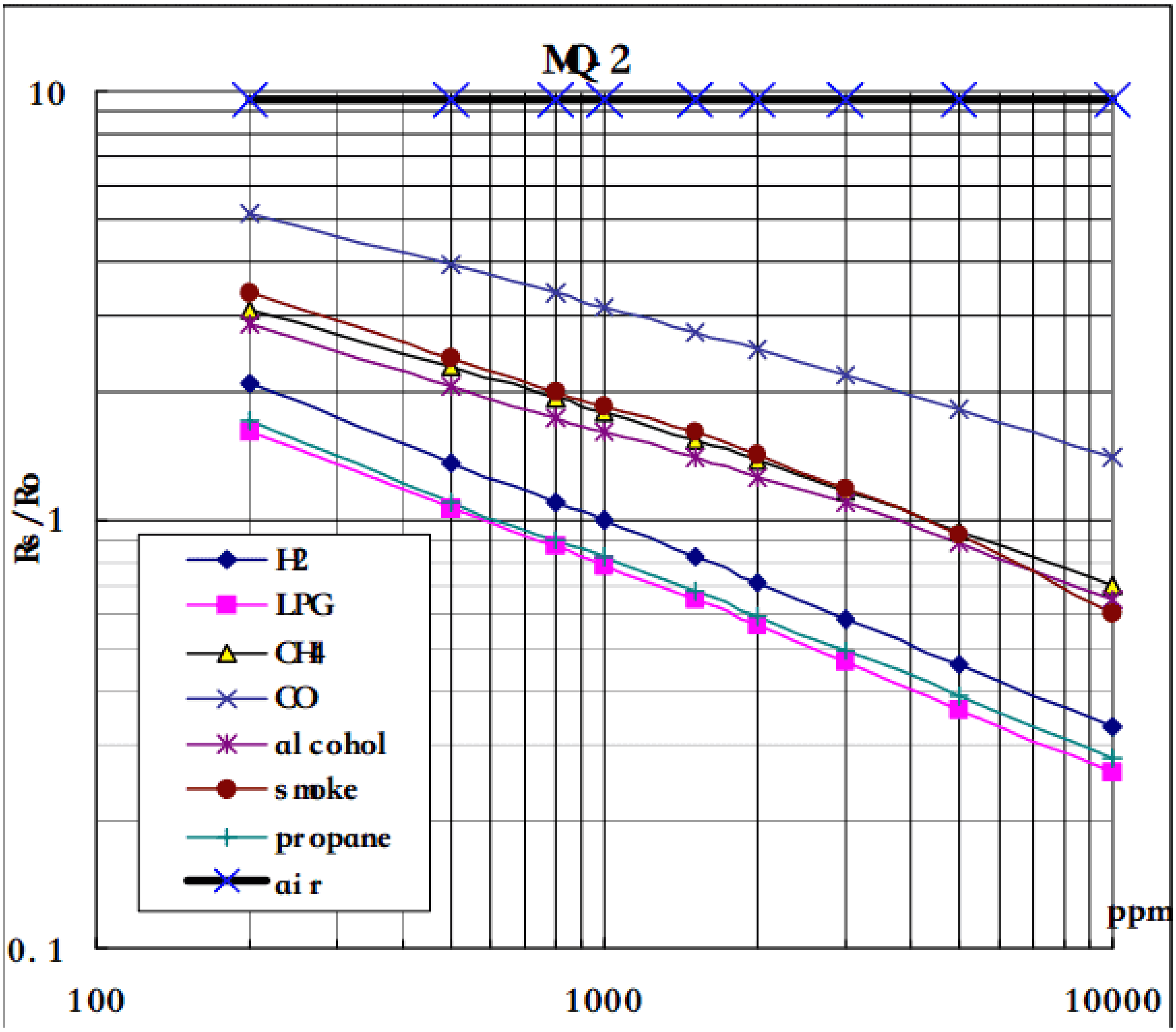
From the previous chapter you already know that the MQ2 gas sensor is sensitive to the gases: methane, butane, LPG and smoke in general. These gases can be measured in concentrations between 300ppm and 10,000ppm.

Because it takes some time until the surface is hot, the MQ2 gas sensor has a preheat duration of around 2 minutes and has a current consumption of 88mA also during the preheat duration. With a 5V supply voltage, the MQ2 gas sensor has a power consumption of 440mW.

How does the MQ2 Gas Sensor measure the Gas?

The MQ2 gas sensor module is only suitable to measure a trend of gas concentration and not the exact gas concentration because the relation between ration in voltage change and concentration is nonlinear. If you want to measure the exact concentration, you have to buy a much more precise and costly sensor.

The following picture shows the minimum and maximum concentration of different gases that can be measured with the sensor.



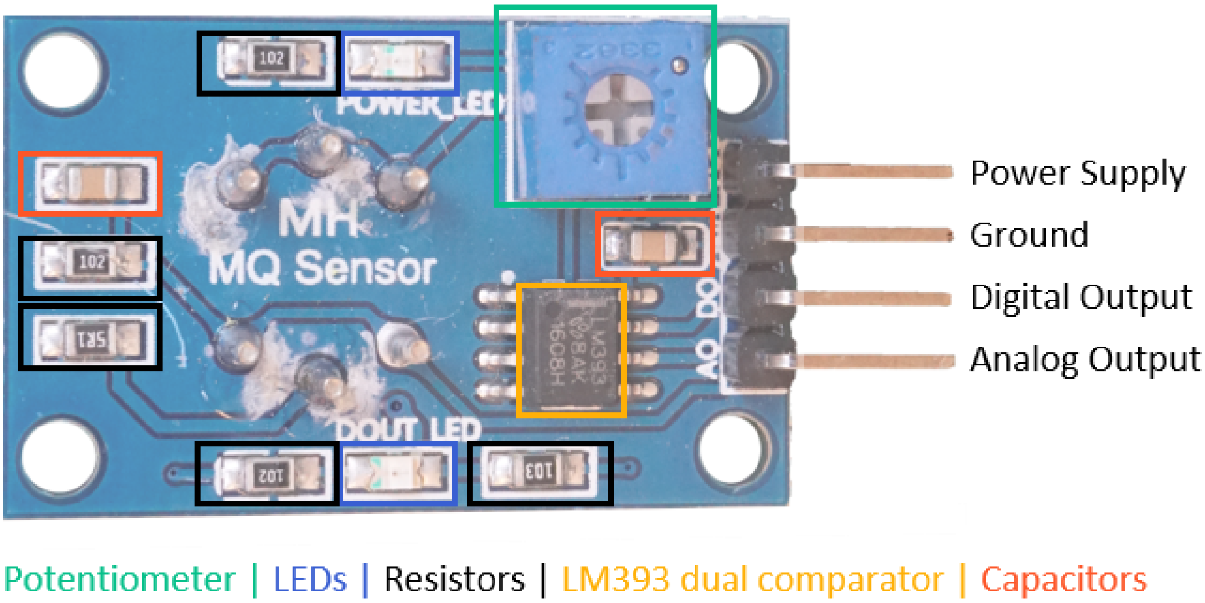
Source picture

According to the graph, you see that the minimum gas concentration that can be measured is around 100ppm and the maximum around 10000ppm. The unit ppm stands for parts per million and therefore the concentration of gas between 0.01% and 1% can be measured.

To understand the working principle of the MQ2 gas sensor module in general, the graph shows that **the resistance of the gas sensor decreases with an increasing gas concentration**.

MQ2 Gas Sensor Module Pinout and Electronic Components

The following picture shows the back side of the MQ2 gas sensor breakout board.



The MQ2 gas sensor module consists of the following electronic parts:

- **4 output pins** that connect the gas sensor to a microcontroller.
  - A0: Analog pin to transfer an analog signal.
  - GND: Ground to connect the gas sensor to ground with the microcontroller.
  - VCC: Pin for the 5V operation voltage regarding to the technical datasheet.
  - D0: Digital output based on a predefined threshold through the potentiometer and the operation voltage of the microcontroller.
- **Potentiometer** to define a threshold for the digital output pin.
- **2 LEDs** to indicate that the module is operating (POWER_LED) and to indicate the status of the digital pin (DOUT_LED).
- **5 Resistors** to prevent LEDs for too high voltages and to

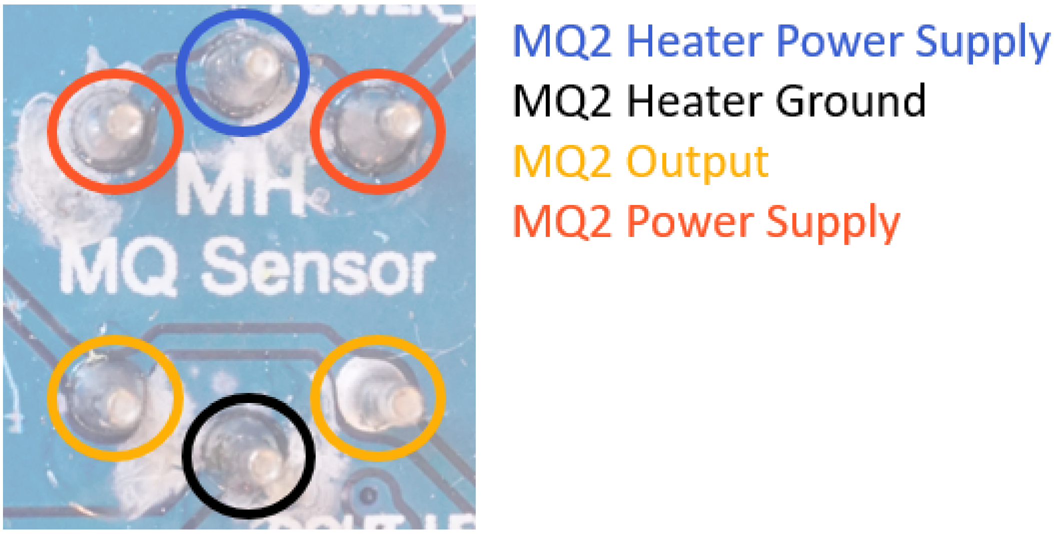
operate as voltage dividers.

- **LM393 dual comparator** to compare the signal created by the MQ2 gas sensor with the predefined value through the potentiometer and to control the status of the LED that indicates the status of the digital output.
- **2 Capacitors** to filter the voltage and stabilize the input and output.

In the next sub-chapter, we take a look at the MQ2 gas sensor pinout but you could also add every other gas sensor to the sensor module when the pinout and geometry of the sensor is the same.

MQ2 Gas Sensor Pinout

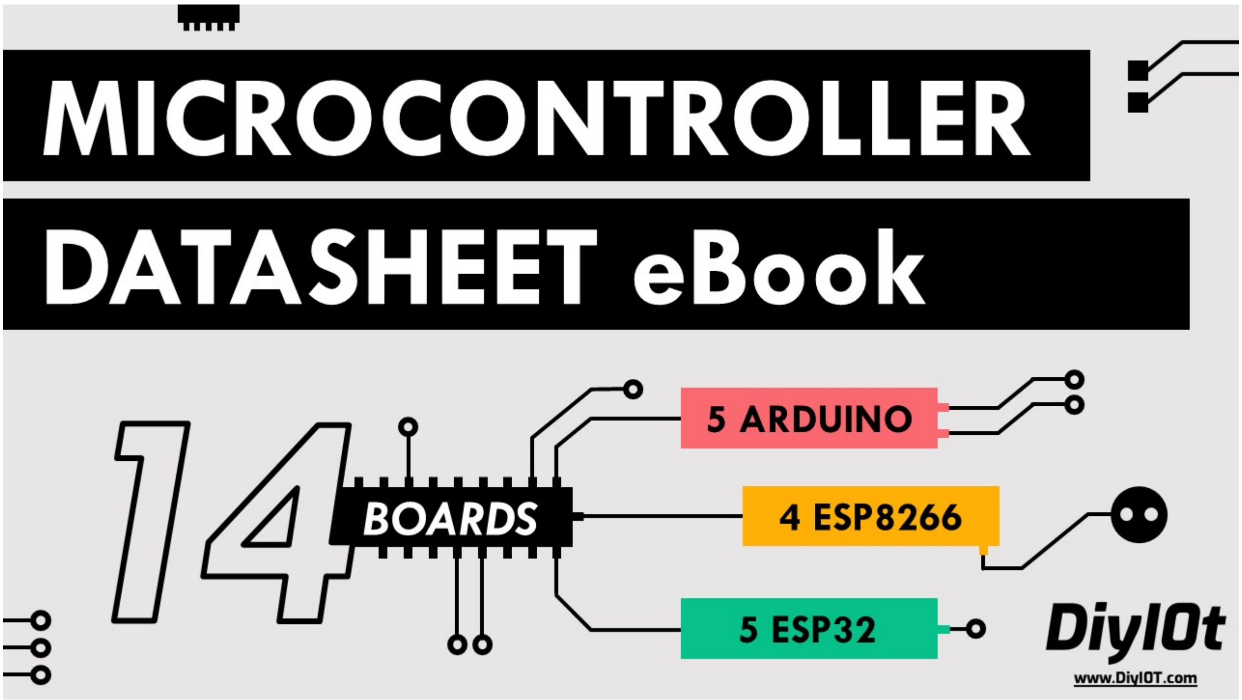
The gas sensor itself is the central part of the MQ2 gas sensor module. The following picture shows the pinout of the MQ2 gas sensor.



The MQ2 gas sensor has in total 6 pins with the following functionality:

- MQ2 Heater Power Supply (1): Power supply for the heater of the SnO2 surface. For my gas sensor module, the pin is also connected to VCC, but it could be an independent second power supply.

- MQ2 Heater Ground (1): Ground to close the circuit of the heater.
- MQ2 Output (2): Output of the MQ2 sensor that is connected to the circuit of the gas sensor module.
- MQ2 Power Supply (2): Power supply for the MQ2 sensor and connected to VCC.



Microcontroller Datasheet eBook

The **35 pages** Microcontroller Datasheet Playbook contains the most useful information of **14 Arduino, ESP8266 and ESP32** microcontroller boards.

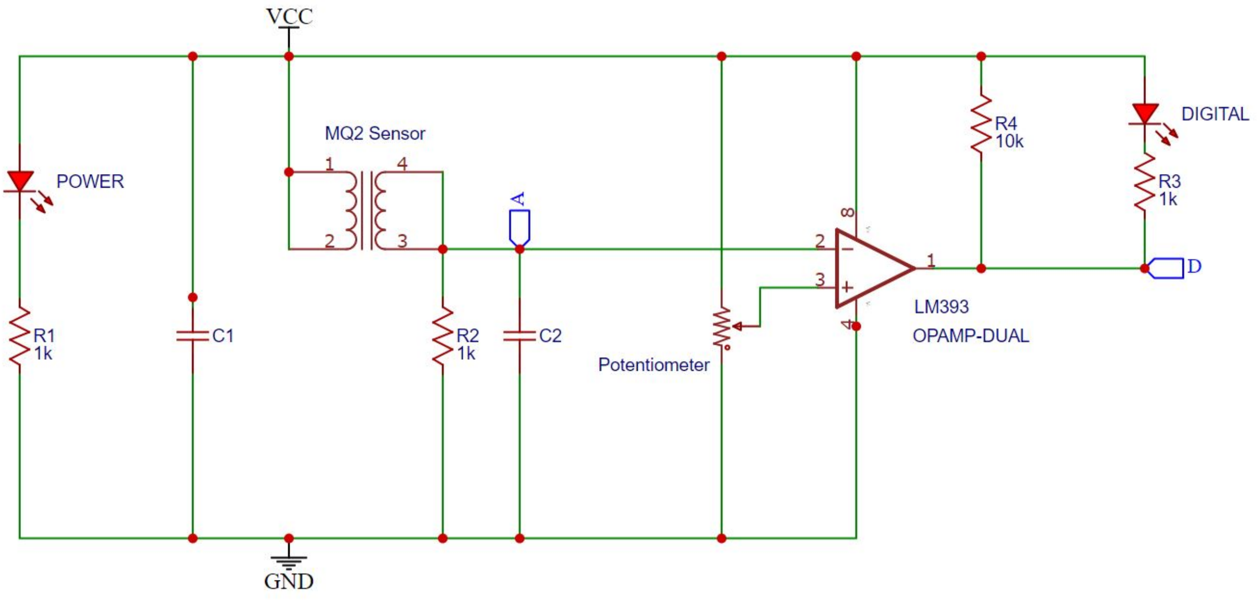
Get more Information

Schematic of the MQ2 Gas Sensor Module

To understand how the MQ2 gas sensor module is working and how the analog and digital output changes when the MQ2 gas sensor is measuring gas, we must take a look at the schematic of the sensor module.

The schematic of the MQ2 gas sensor module shows us how all electronic parts of the module that we discussed in the previous sub-chapter are connected.

The following picture shows the schematic of the MQ2 gas sensor module.



On the left side we see the LED (POWER_LED) for the power indication in series with the resistor R1 (1kΩ) to limit the voltage for the LED. If the gas sensor module has a valid power supply, this LED turns on independent of the gas sensor itself.

The capacitor C1 is also connected between VCC and GND to stabilize the operation voltage and would not be necessary if you use a stable power supply like the Arduino, ESP8266 or ESP32 microcontroller board with 5V voltage regulator.

The MQ2 gas sensor is connected with both MQ2 Power Supply pins to the external power supply. The output of the MQ2 gas sensor build with resistor R2 (1kΩ) a voltage divider. The voltage on the output of the gas sensor is also connected to the analog output of the sensor module (A),

the capacitor C2 and is also the reference voltage of the LM393 comparator (2).

The input voltage of this comparator (3) is the output of the potentiometer that acts like an adjustable voltage divider.

The output of the LM393 dual comparator (1) is connected to the digital output of the MQ2 gas sensor module (D) and connected to the circuit part with resistor R4 (10kΩ) as well as LED (DOUT_LED) and resistor R3 (1kΩ).

Functionality of the MQ2 Gas Sensor Module

Functionality of the MQ2 Gas Sensor Module

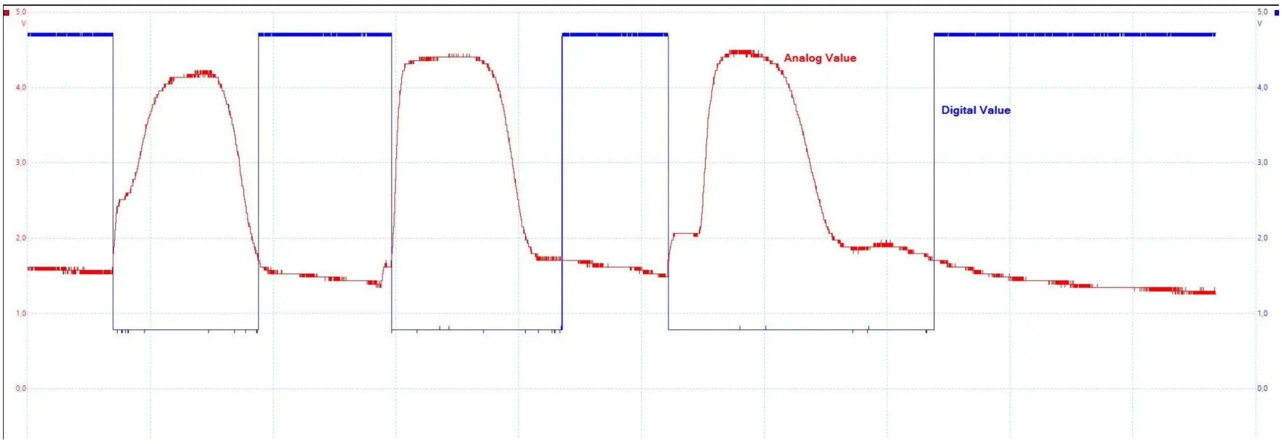
After we know the schematic of the MQ2 gas sensor module, the functionality is not hard to understand. We already know from the picture of the gas measurement that **the resistance of the MQ2 gas sensor decreases when the gas concentration that is measured increases**.

If the gas concentration increases, the voltage over the MQ2 sensor decreases due to the decrease of resistance (U=R*I). Because the voltage between VCC and GND must be the same, the voltage over resistance R2 increases and therefore the potential on the point where the analog value is measured. **The analog sensor value increases due to the increase of gas concentration**.

The potentiometer must be calibrated in a way that if there is no relevant gas concentration, the input voltage of the comparator (3) has to be higher than the output voltage of the MQ2 sensor that equals the reference voltage of the comparator (2). In this case, the output of the comparator is equal to VCC and therefore also the digital output of the sensor module. Because there is no potential difference between resistor R3 and LED (DOUT_LED), the LED is off.

If the gas concentration rises, the reference voltage (2) rises and should exceed the input voltage of the comparator (3). In this case, the output of the comparator switches from VCC to GND and therefore also the digital output. Now there is a potential difference between resistor R3 and LED (DOUT_LED), the LED is on.

The relationship of the analog and digital output is also shown in the following picture, where I measured the analog and digital output of the MQ2 gas sensor module with my oscilloscope. Click on the image to enlarge it.



The following pullet points summarize the functionality of the MQ2 gas sensor module.

- If the measured gas concentration increases, the analog output also increases.
- To increase or decrease the sensitivity of the digital output of the MQ2 gas sensor module, adjust the potentiometer.
- If the threshold of the potentiometer is not exceeded, the digital output is HIGH and the LED (DOUT_LED) is off.
- If the gas concentration increases further and the threshold of the potentiometer is exceeded, the digital output is LOW and the LED (DOUT_LED) is on.

The following table gives you an overview of all components and parts that I used for this tutorial. I get commissions for purchases made through links in this table.

Component	Amazon Link	AliExpress Link
Arduino Nano	Amazon	AliExpress
Arduino Pro Mini	Amazon	AliExpress
Arduino Uno	Amazon	AliExpress
Arduino Mega	Amazon	AliExpress
ESP32 ESP-WROOM-32	Amazon	AliExpress
ESP8266 NodeMCU	Amazon	AliExpress
ESP8266 WeMos D1 Mini	Amazon	AliExpress
MQ2 Gas Sensor (in Sensor Kit)	Amazon	AliExpress
Active Buzzer in Sensor Kit	Amazon	AliExpress

Building a Gas Alarm with Arduino, ESP8266 or ESP32

Now we want to build a gas alarm with the MQ2 gas sensor, an active buzzer and your favorite Arduino, ESP8266 or ESP32 microcontroller. The following pictures show how to connect the MQ2 gas sensor to different microcontroller boards. I also added an active buzzer to the project that makes a sound if the sensor detects gas in the air. If you are interested in more details about active or passive buzzers, visit the buzzer tutorial.

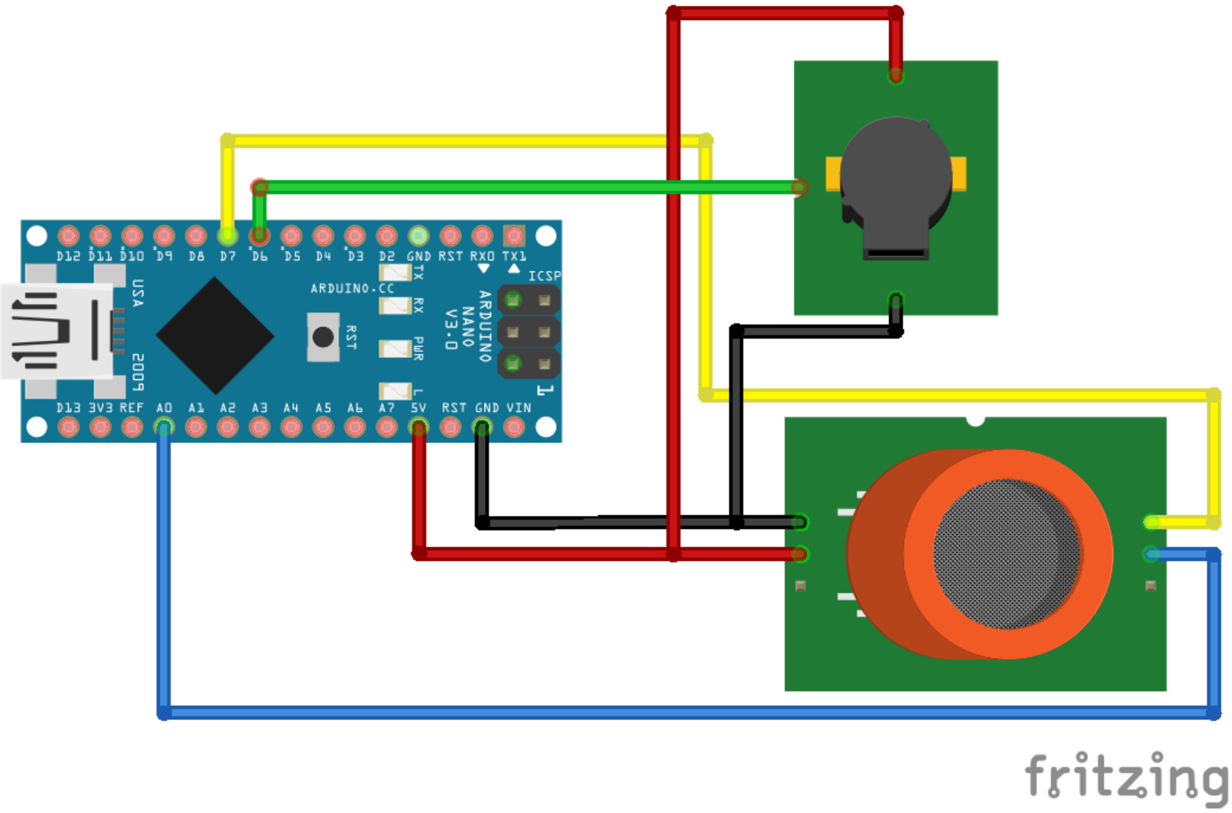
If your favorite board is missing in the following wiring sections, use the comment section below and I will add your board to this article.

Wiring between MQ2 Gas Sensor and Arduino Boards

The operation voltage of the MQ2 gas sensor is 5V. Almost every Arduino microcontroller has an operation voltage of 5V but the Arduino Pro Mini has a version with an operation voltage of 5V and also of 3.3V. Therefore you have to use the 5V version of you want to use the MQ2 gas sensor.

The following picture show the wiring between different Arduino boards, the MQ2 gas sensor and the active buzzer.

Arduino Nano


D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
ARDUINO.CC
ARDUINO NANO V3.0
ICSP
USA
2009
RST
TX
RX
PWR
L
fritzing

MQ2 Gas Sensor Arduino Nano

For more information about the Arduino Nano, visit the Arduino Nano Tutorial.

Arduino Pro Mini 5V

Arduino Uno

Arduino Mega

Wiring between MQ2 Gas Sensor and ESP8266 Boards

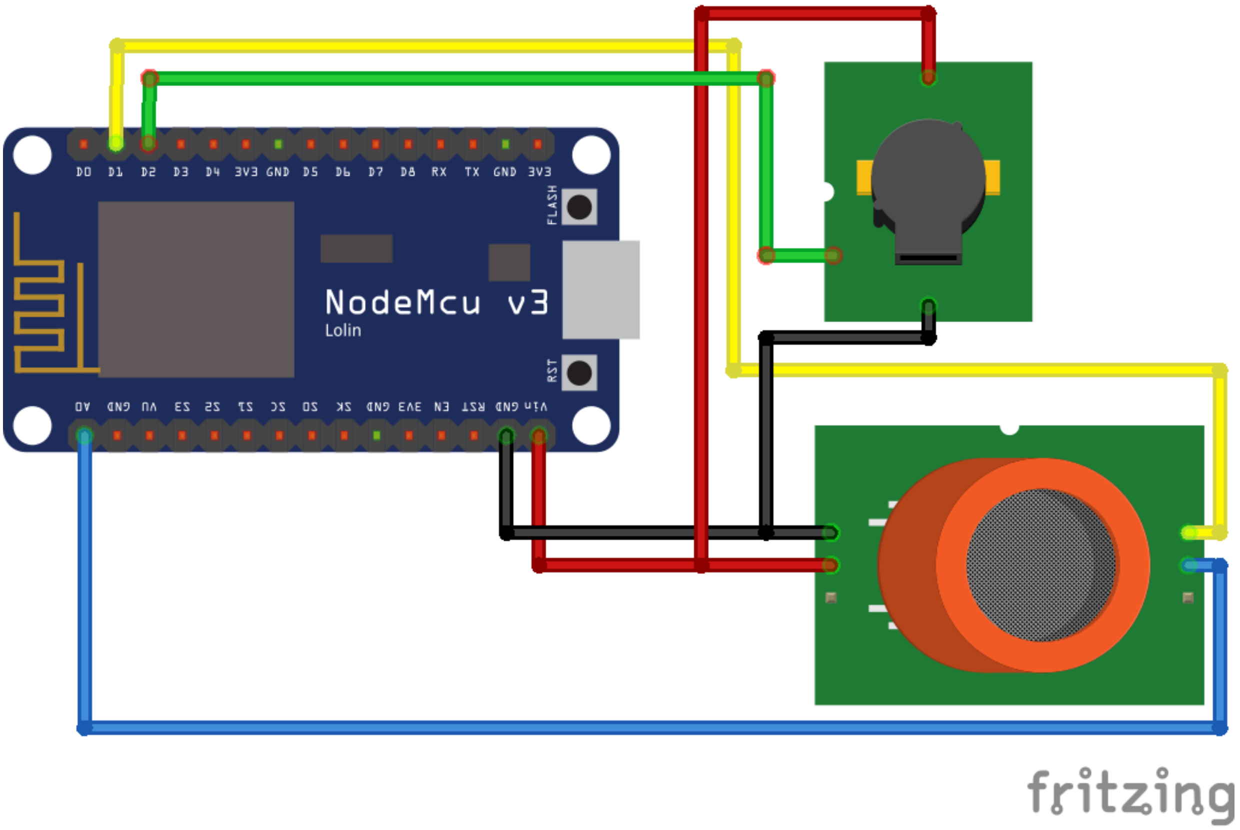
The ESP8266 microcontroller has an operation voltage of 3.3V that is too low for the requiring operation voltage of 5V for the MQ2 gas sensor.

If the ESP8266 board is powered via 5V USB, the VIN pin of the ESP8266 NodeMCU and the 5V pin of the ESP8266 WeMos D1 Mini

ESP8266 NodeMCU and the 5V pin of the ESP8266 WeMos D1 Mini have an output voltage slightly lower than 5V that is enough to supply the MQ2 gas sensor.

The following pictures show the wiring between the MQ2 gas sensor, the active buzzer and the ESP8266 NodeMCU as well as the ESP8266 WeMos D1 Mini.

ESP8266 NodeMCU


D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
NodeMcu v3
Lolin
RST
A0 GND VU S3 S2 S1 SC S0 SK GND 3V3 EN RST GND Vin
fritzing

MQ2 Gas Sensor ESP8266 NodeMCU

ESP8266 WeMos D1 Mini

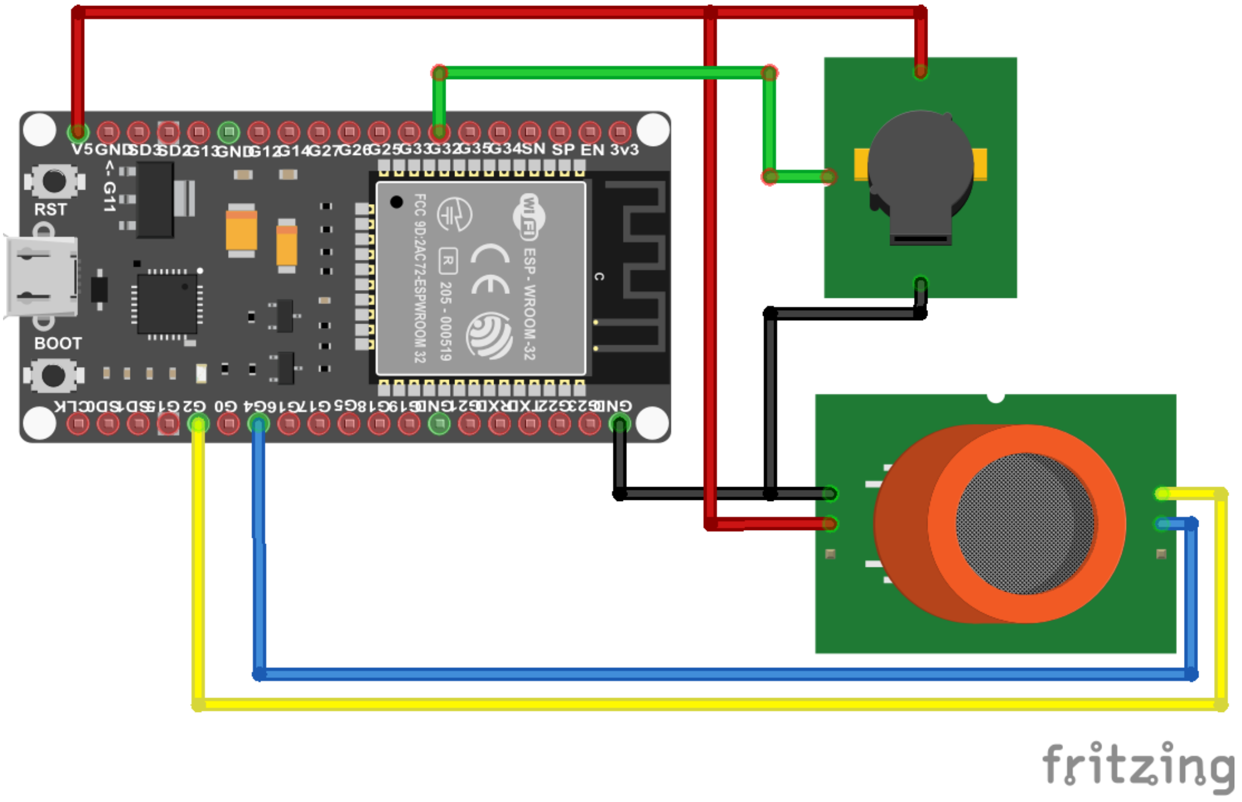
Wiring between MQ2 Gas Sensor and ESP32 Boards

Like the ESP8266, the ESP32 the operation voltage of the ESP32 is 3.3V and therefore too low for the MQ2 gas sensor, that has an operation voltage of 5V.

But if we power the ESP32 board via USB, the input voltage of the board before the voltage regulator is 5V and accessible via the V5 pin of the EPS32 ESP-WROOM-32. If you did not know the voltage levels of different ESP32 boards, I recommend my Microcontroller Datasheet eBook, where you find all these information.

The following picture shows the wiring between the EPS32 ESP-WROOM-32, the MQ2 gas sensor and the active buzzer.

EPS32 ESP-WROOM-32



MQ2 Gas Sensor ESP32 NodeMCU

Program Code for a Gas Alarm with Arduino, ESP8266 or ESP32

From the wiring between the MQ2 gas sensor and the different Arduino, ESP8266 and ESP32 microcontroller, you see that I use different pins. Therefore we have to take care, that the right pins are used in the program code.

```
//int Buzzer = 6;        // used for Arduino
//int Gas_analog = A0;    // used for Arduino
//int Gas_digital = 7;   // used for Arduino

//int Buzzer = D2;        // used for ESP8266
//int Gas_analog = A0;    // used for ESP8266
//int Gas_digital = D1;   // used for ESP8266

int Buzzer = 32;        // used for ESP32
int Gas_analog = 4;    // used for ESP32
int Gas_digital = 2;   // used for ESP32

void setup() {
  Serial.begin(115200);
  pinMode(Buzzer, OUTPUT);
  pinMode(Gas_digital, INPUT);
}

void loop() {
  int gassensorAnalog = analogRead(Gas_analog);
  int gassensorDigital = digitalRead(Gas_digital);

  Serial.print("Gas Sensor: ");
  Serial.print(gassensorAnalog);
  Serial.print("\t");
  Serial.print("Gas Class: ");
  Serial.print(gassensorDigital);
  Serial.print("\t");
  Serial.print("\t");
  Serial.print("\t");

  if (gassensorAnalog > 1000) {
    Serial.println("Gas");
    digitalWrite (Buzzer, HIGH) ; //send tone
    delay(1000);
    digitalWrite (Buzzer, LOW) ;  //no tone
  }
  else {
    Serial.println("No Gas");
  }
  delay(100);
}
```

In the first part of the program code we define the pins that connect the MQ2 gas sensor, the active buzzer and the Arduino, ESP8266 or ESP32 microcontroller.

The following table shows what pins I choose for the connection.

Pin			
Power Supply	5V	VIN	V5
Ground	GND	GND	GND
Digital pin to active buzzer	6	D2	32
Analog pin from MQ2 gas sensor	A0	A0	4
Digital pin from MQ2 gas sensor	7	D1	0

Because I added the lines of code for all three microcontrollers, you

Because I added the lines of code for all three microcontrollers, you have to comment and uncomment the first part of the code dependent

on your microcontroller.

The ESP32 has a build-in analog to digital converter, so you can use multiple pins as analog input. The pins that you can use as analog pins are described in the ESP32 pinout tutorial.

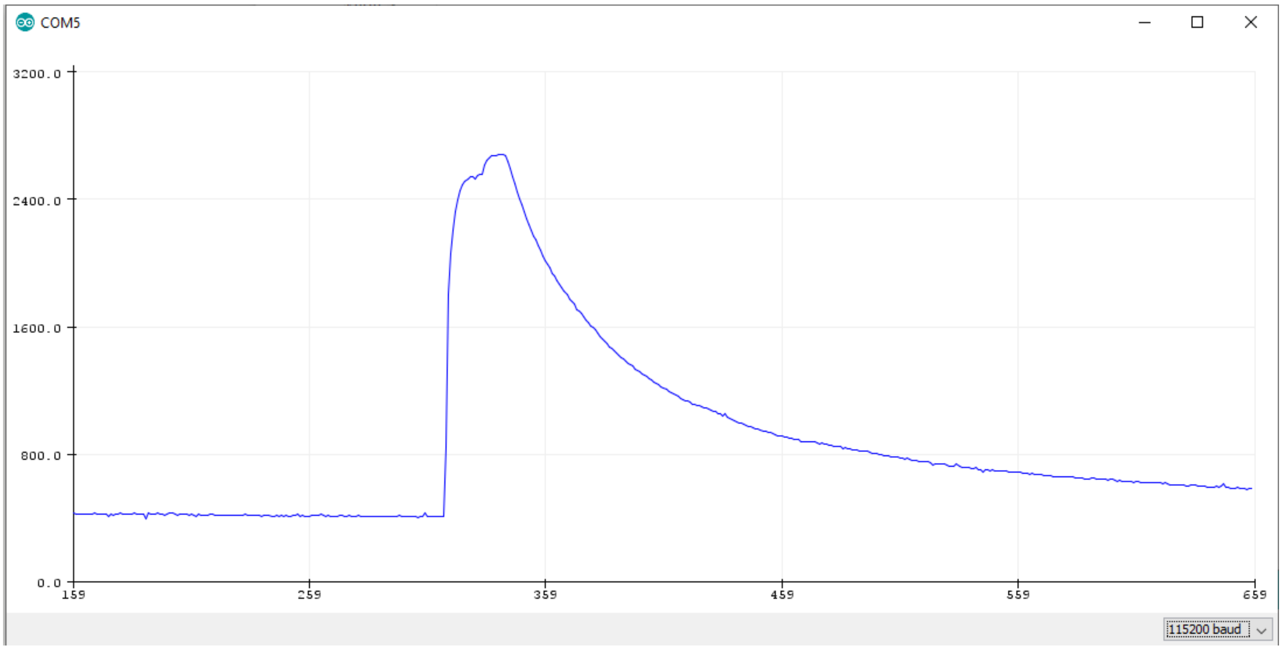
In the setup function we set the baud rate to 115200 and define the pin modes:

- The microcontroller sends the digital value to the buzzer → Output
- The microcontroller gets the current digital status of the MQ2 gas sensor → Input

In the loop function we read the analog and digital value of the MQ2 gas sensor and save the values in different variables. Now we print the values to the serial output to see the values in the serial monitor of the Arduino IDE. For a better reading we separate each variable with a tabulator.

Now we have to define the threshold when we want to activate the buzzer or not. Therefore we compare the current analog sensor value with a constant, in my case 1000. If the analog sensor is greater than 1000, gas is detected and we set the buzzer active for one second. After this one second we deactivate the active buzzer again.

The following picture shows the output of the serial monitor with the program code in the next section.



You see clearly, when I activate the cigarette lighter, the analog value spikes above 1000. Also in the following video you see that when the gas sensor detects the gas, the buzzer is activated.

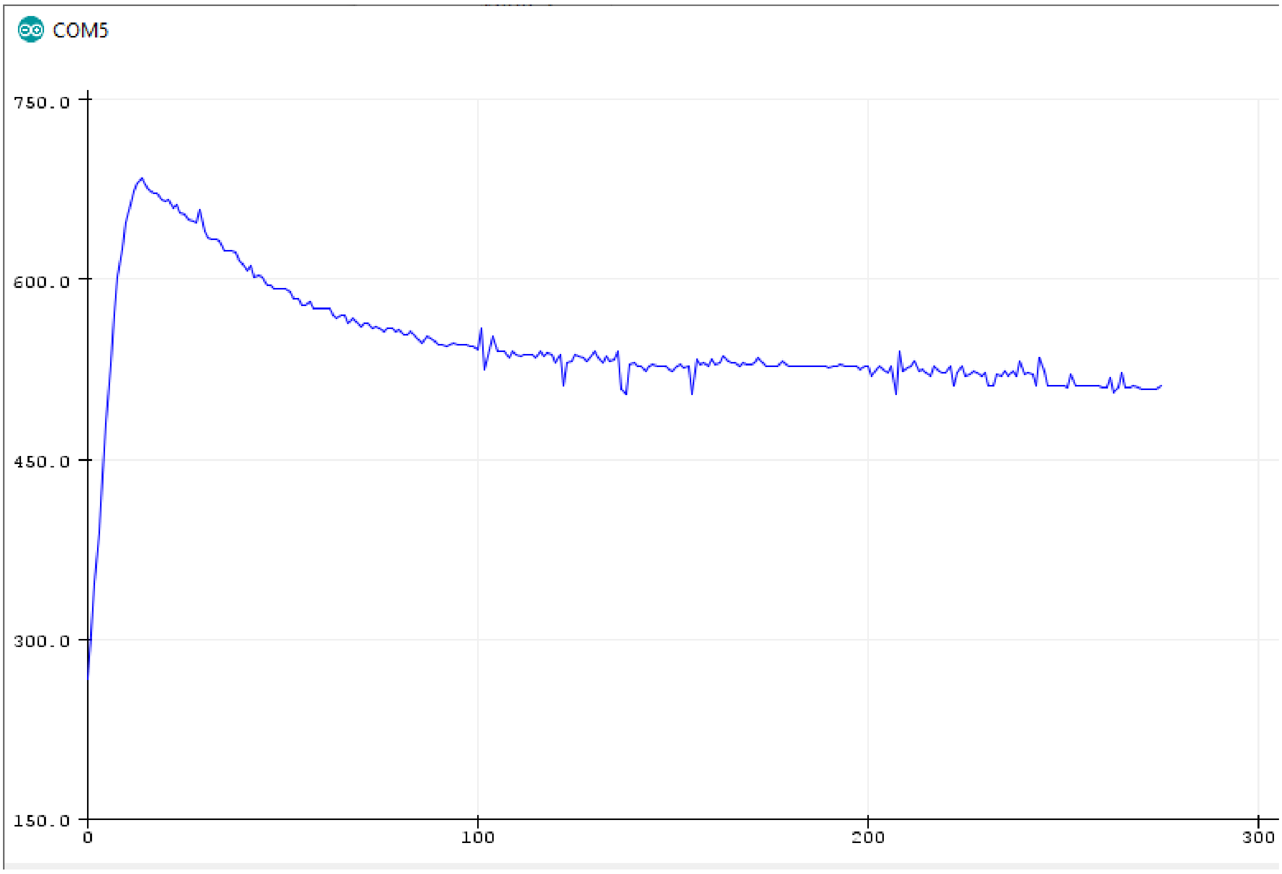
Bonus: Show Analog Value While MQ2 Preheating

We can also make the preheating visible in the analog signal. The wiring stays the same and you only need the following short program code and open the serial plotter of the Arduino IDE.

```
//int Gas_analog = A0;    // used for Arduino
//int Gas_analog = A0;    // used for ESP8266
int Gas_analog = 4;    // used for ESP32

void setup() {
  Serial.begin(115200);
}

void loop() {
  int gassensorAnalog = analogRead(Gas_analog);
  Serial.println(gassensorAnalog);
  delay(100);
}
```



The x-axes shows the time in 1 seconds. During the preheating the analog sensor value spikes to around 700 in the first 10 seconds and then stabilizes around 500 after 200 seconds.

I hope you enjoyed reading the tutorial about the MQ2 gas sensor. You can also add a flame sensor to the gas alarm to detect an active fire. In the flame sensor tutorial you learn how to add the sensor to your project.

If you have any questions regarding this article, use the following comment section to ask your questions. I will answer them as soon as possible.

Recent Posts

- MH-Z14A CO2 Meter Tutorial with Arduino, ESP8266 or ESP32
- Control a DC Motor with Arduino, ESP8266 or ESP32 without IC
- ESP8266 WeMos D1 Mini Tutorial
- ESP8266 NodeMCU Tutorial
- TCRT5000 Line Tracking Module for Arduino, ESP8266 and ESP32

9 thoughts on “MQ2 Gas Sensor Tutorial for Arduino, ESP8266 and ESP32”

Robert

13. December 2020 at 11:42

Merci beaucoup. Can we use this sensor to make a smoke detector please?

Reply

cdaviddav

11. January 2021 at 05:36

Hi Robert,

what would be the difference between your smoke detector and the Example “Building a Gas Alarm with Arduino, ESP8266 or ESP32”

Reply

Maik

3. March 2021 at 22:01

Isn’t it necessary to map the 5V analog signal of the sensor to 3.3V before feeding it into the ESP32?

Reply

cdaviddav

16. March 2021 at 21:01

Hi Maik,

it is not necessary because if you power the the MQ2 gas sensor via the ESP32, the analog signal has a value range between 0V and 3.3V.

A mapping of the analog signal makes sense if the operation voltage of the sensor module is an external 5V power supply.

Reply

Ludvig

29. March 2021 at 22:12

The standard test conditions for the MQ sensors is 5V+-0.1V, so powering it with 3.3V will could give you unreliable readings. If you need precise output, you should consider using a 5V power source, regulate it down to 3.3V for your MCU (this is included in most devkits) and keep 5V for the sensor. Then use a voltage divider to convert the 0-5V signal to 0-3.3V.

If you only need a digital signal, the details might not be so important and using 3.3V could be enough. Of course it depends on you application and the reliability requirements you have.

Reply

Rob

21. June 2021 at 22:53

But you’re not powering with 3.3v on the esp32, you are using the 5v pin (when powered with USB) which is what the sensor needs (4.5v-5v). If you want to measure the analog signal with an esp32 you will need a voltage divider at the very least to scale the 0v-5v analog sensor value down to 0v-3.3v to the ADC of the esp32. Depending on the impedance of the sensor, an Op Amp used as a voltage buffer may be necessary in addition to the voltage divider.

Reply

Oleg Boldyrev

15. March 2021 at 20:34

Hi Christopher,

Thanks for the excellent tutorial.

Did you mean something special by depicting the sensor as a transformer or was it an idiom? Does the numbering of pins on the schematic correlate to the one on the picture of the sensor?

Reply

cdaviddav

16. March 2021 at 20:56

Hi Oleg,

thanks for the feedback. In the software for the schematic, I needed a part that has 4 pins and the transformer was nearly the only one. Therefore the transformer for the MQ2 Sensor is only a symbol. Yes, the pins are correlate to the ones on the picture.

Reply

Ludvig

29. March 2021 at 22:05

Hi! I am trying to compute the Rs value (the resistande over the sensor). I do get approximately correct values if I build a voltage divider and measure the voltage drop with a multimeter, but the calculations only work out if I remove the resistor R2 from the calculations. The schamatic you have provided seems correct from my own examination of the board, but I cannot make heads or tails from my multimeter readings… Do you have any data sheet that can verify your schematic?

Reply

Leave a Comment

Name

Email

Website

Post Comment

As an Amazon Associate I earn from qualifying purchases.